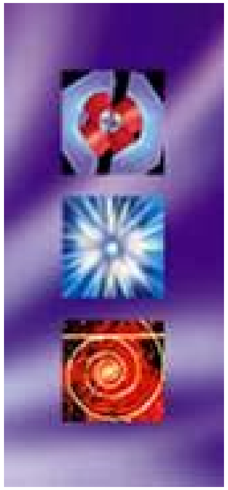
Numerical Optimisation Methods
An Introduction

Hywel Owen

Daresbury Laboratory



What's in the talk

- Optimisation - some definitions
- Traditional optimisation methods - minimisation and all that
- Recent methods - annealing and genetics

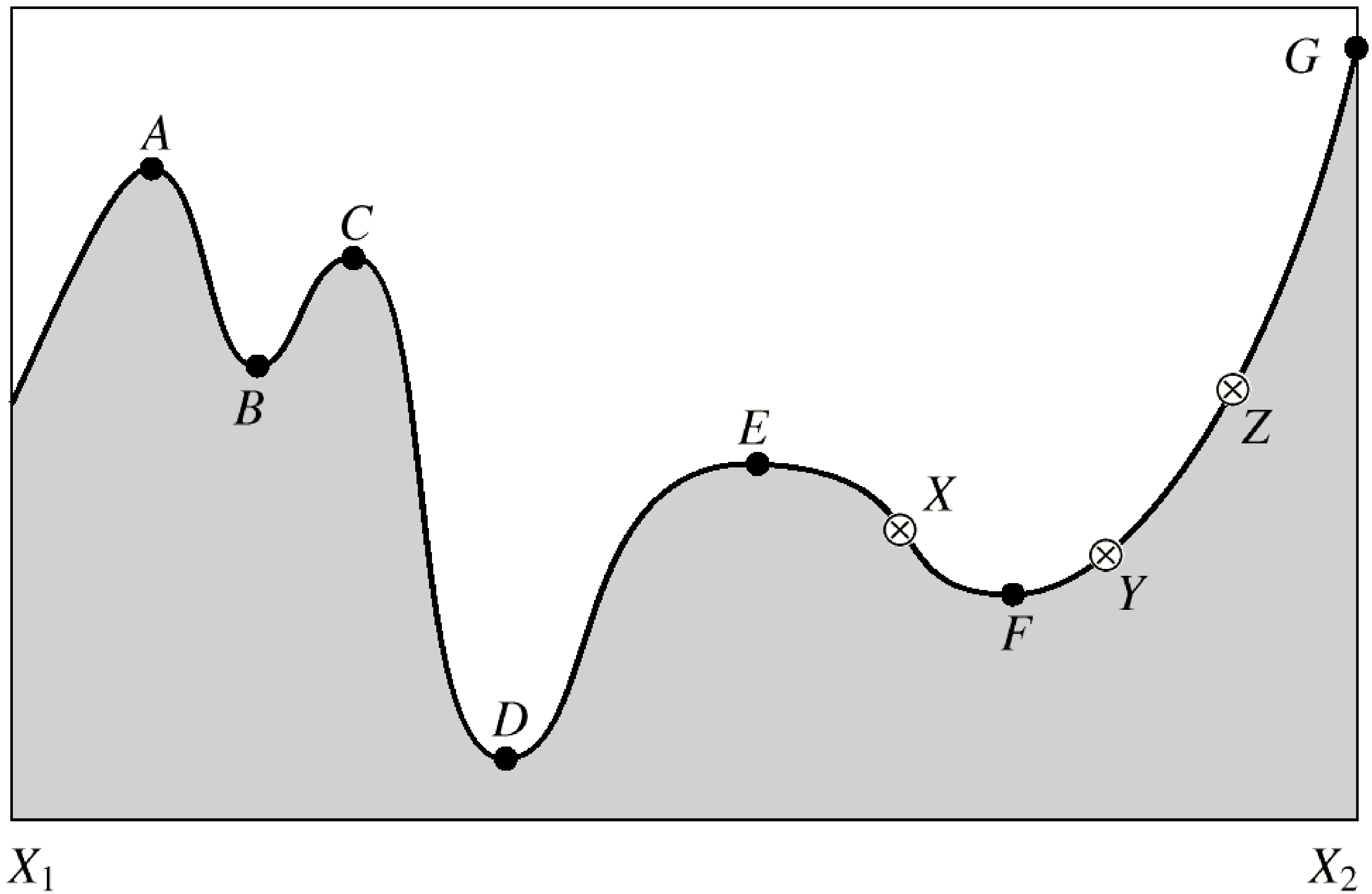
The design process

- Design is creating something to fit a purpose - from toothbrushes to accelerators.
- A design is judged to be good by *quantifying* how good it is compared to other designs.
- The space of possible designs is termed the *Configuration Space*.
  - Generally quantifiable via some variables.
- The ‘goodness’ of the design is termed the *Objective Function*.
  - Must be quantifiable if we wish to do an optimisation, but can be a ranking scheme.

Optimisation

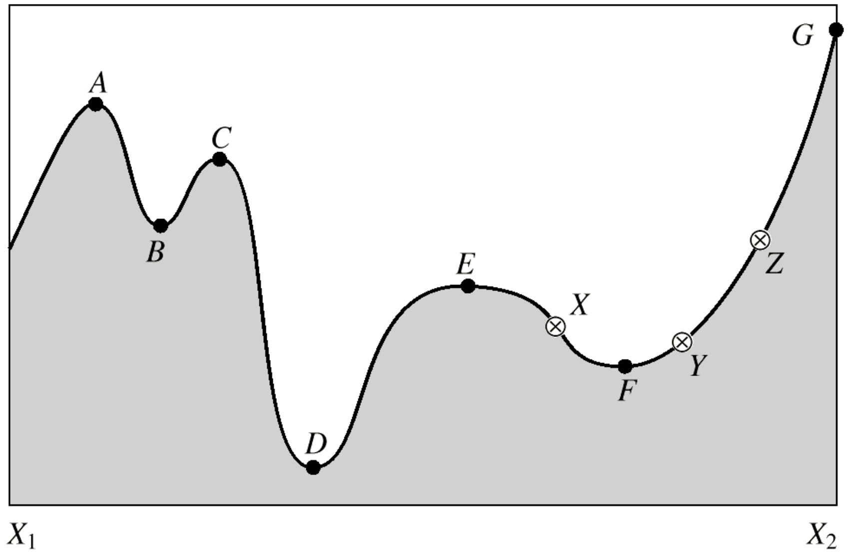
- *Optimisation* is the improving of a design. This means either maximising an Objective Function *F*, or equivalently minimising - *F*
- Division between *Constrained* and *Unconstrained*, and *Discrete* and *Continuous*.
- Constrained Optimisation means that the configuration space is divided into *Feasible* and *Non-Feasible* solutions.
- Strong link with mathematics of functions.

One Dimension (One Variable)

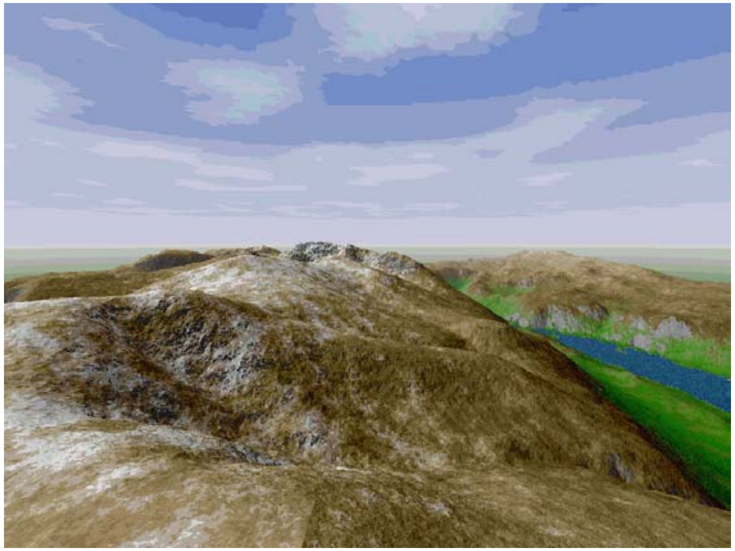


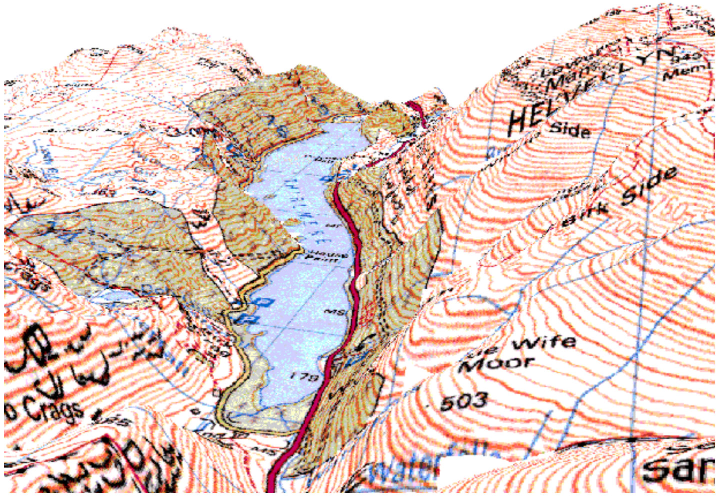
One Dimension

- Analogous to root-finding methods, e.g. d*F*/dz=0.
- Many methods - Secant, Brent's, Newton-Raphson etc. Some (e.g. Newton-Raphson) require calculation of the local gradient of the function.
- All require the Objective Function to be reasonably well-behaved, e.g. smooth and roots reasonably far apart.



Multiple Dimensions

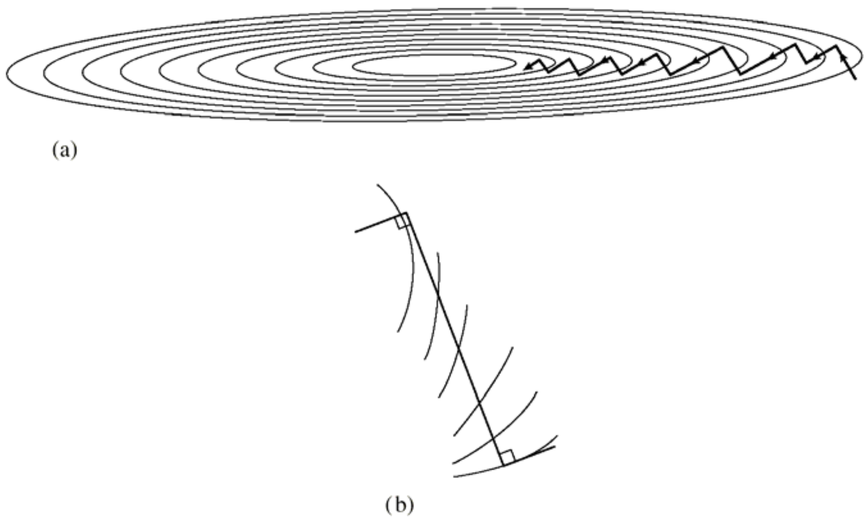




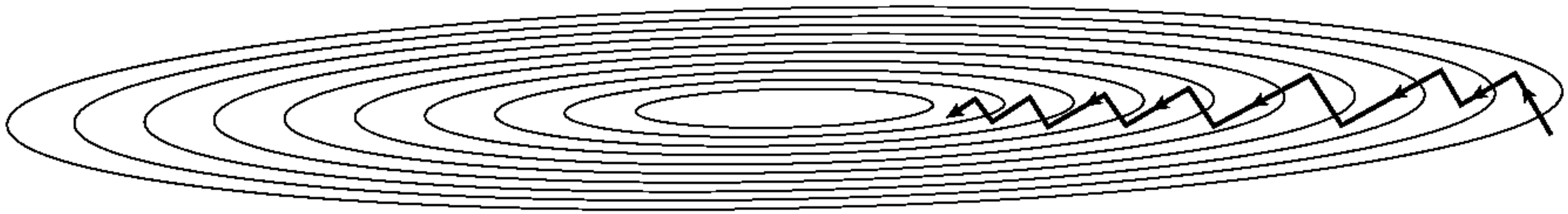
- The objective function can be thought of as a surface in two dimensions.
- Higher dimensions can be thought of in an analogous way.

Cauchy Method of Steepest Descent

- Requires that the local gradient of the objective function F can be calculated in some way
- Choose point $\underline{P}_0$
- Move from $\underline{P}_i$ to $\underline{P}_{i+1}$ by minimising along the direction $-\nabla F$
- Use conjugate gradient method to reduce number of steps



Cauchy Method of Steepest Descent



(a)

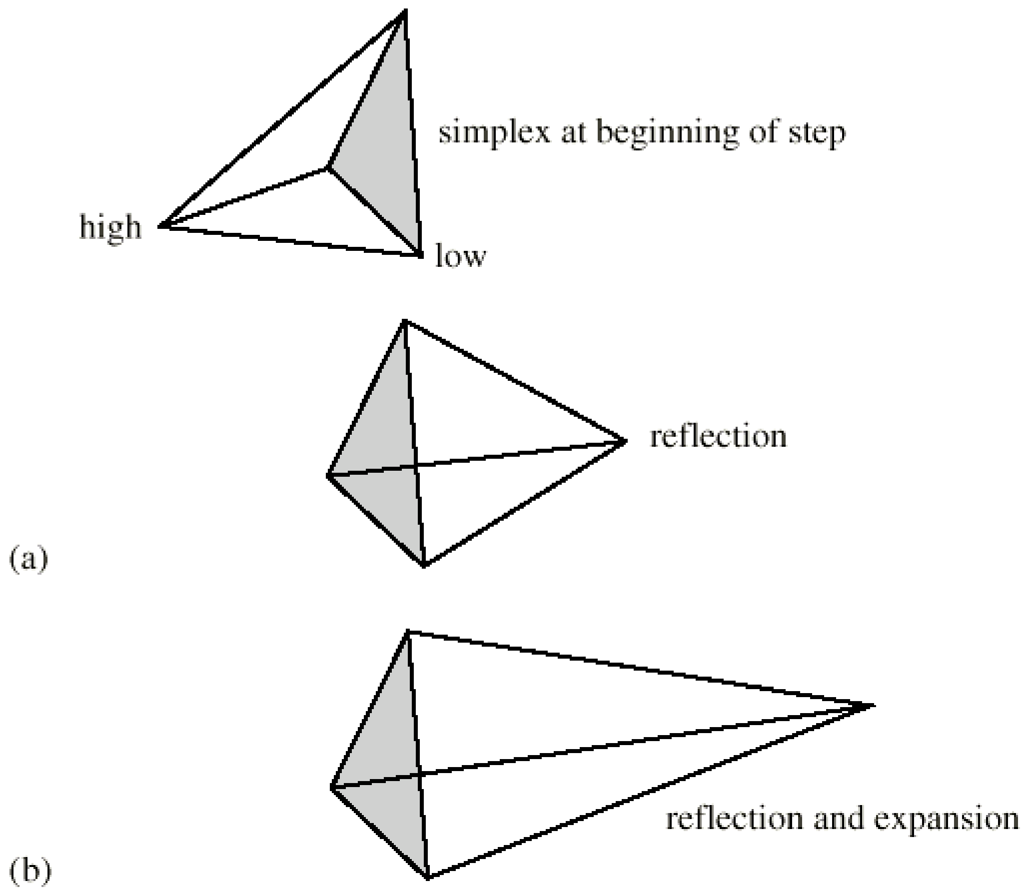
(b)

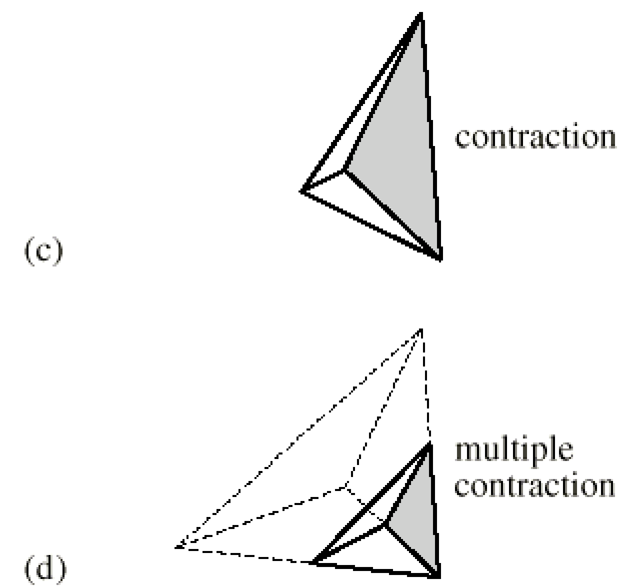
The Downhill Simplex Method (Nelder & Mead, 1965)

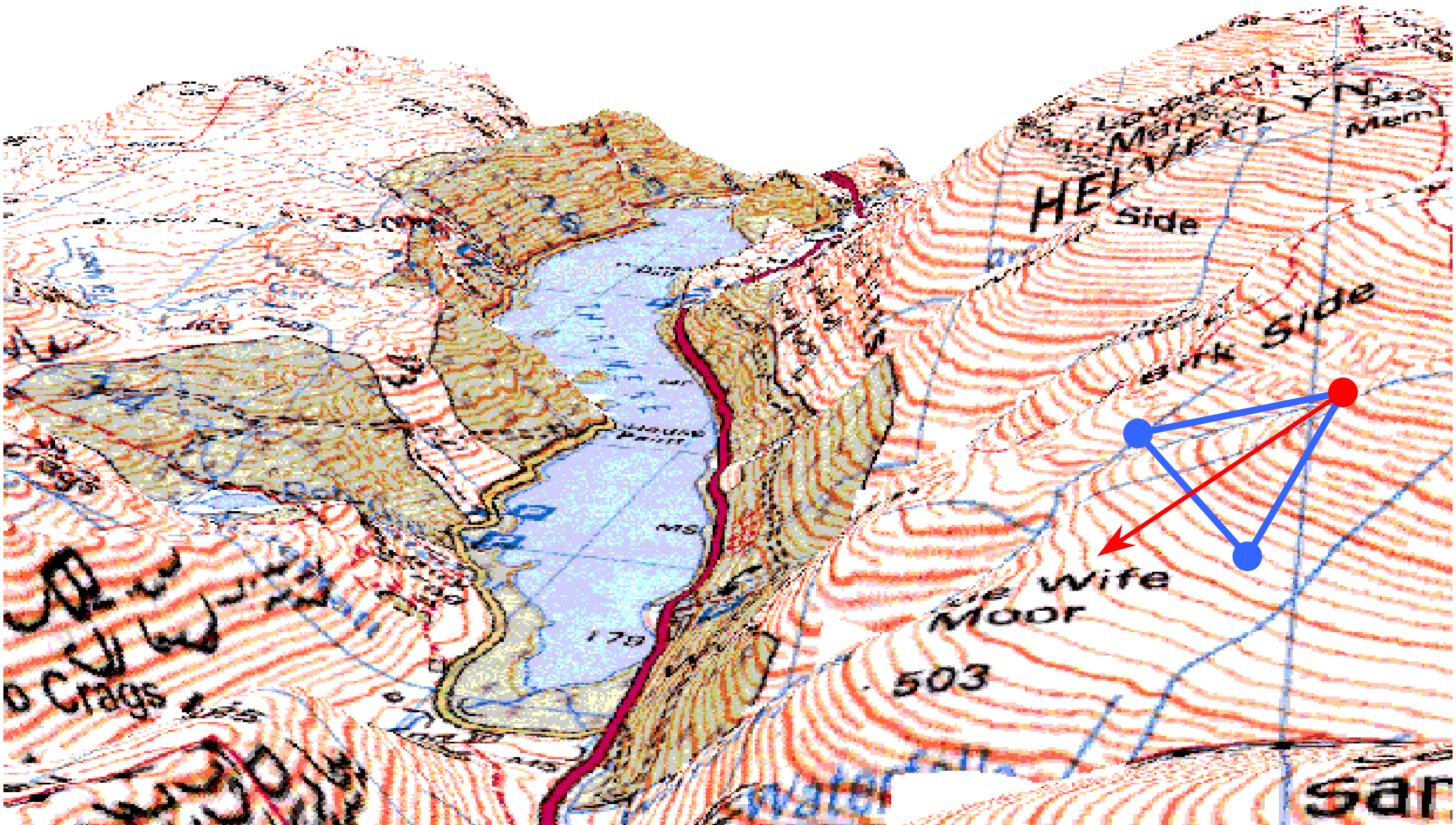
- Simplex - geometrical figure in n dimensions, with $n+1$ vertices.
  - Triangle in 2 dimensions, tetrahedron in 3 dimensions…
- Choose starting point $\underline{P}_0$, and create simplex by adding each of the unit vectors $\underline{e}_i$ for each vertex.
- Evaluate F for each vertex. Choose new simplex.

The Downhill Simplex Method (Nelder & Mead, 1965)

Scaling values are used to choose the expansion and contraction rates






Side
Side
Wife
Moor
503
Crags

Powell's Direction Set Method (Powell, 1964)

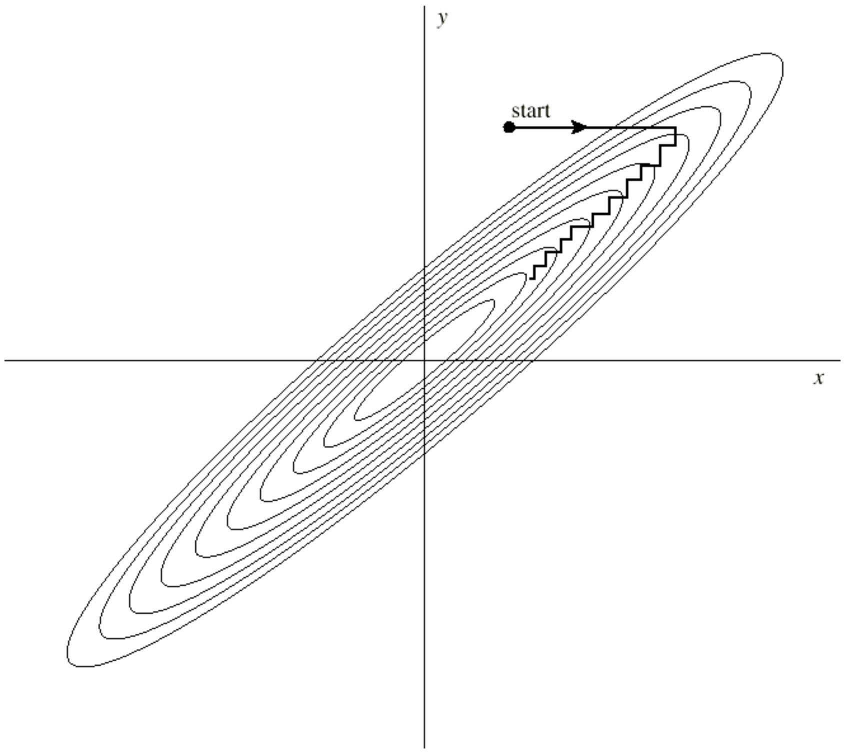
- Choose starting point $\underline{P}_0$, and set of direction vectors $\underline{u}_i = \underline{e}_i$ - e.g. each of the parameters

- Along each $\underline{u}_i$, minimise the objective function F to give $\underline{P}_i$

- Cycle $\underline{u}_i$ values, e.g. $\underline{u}_i = \underline{u}_i+1$

- Set $\underline{u}_N = \underline{P}_N - \underline{P}_0$

- Move $\underline{P}_N$ to minimum along $\underline{u}_N$ and call it new $\underline{P}_0$



Example - MAD Matching Module

- Objective Function is called Penalty Function, which is minimised. Weighting is accomplished by multiplying the constraint by the weight in the penalty function calculation.
- Three methods used:
  - MIGRAD and LMDIF calculate numerical derivatives of either the penalty function as a whole or of each of the individual constraints
  - SIMPLEX uses the Simplex algorithm.

12.6 Matching Examples

12.6.1 Simple Periodic Beam Line

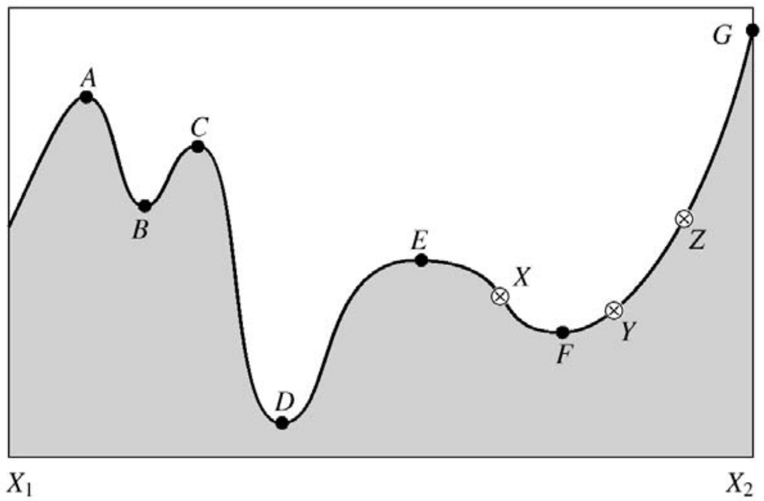
Match a simple cell with given phase advances:

```
QF:     QUADRUPOLE,...
QD:     QUADRUPOLE,...
CELL1:  LINE=(...,QF,...,QD,...)
        USE,CELL1
        CELL
        VARY,NAME=QD[K1],STEP=0.01
        VARY,NAME=QF[K1],STEP=0.01
        CONSTRAINT,PLACE=#E,MUX=0.25,MUY=1/6
        MIGRAD,CALLS=2000
        ENDMATCH
```

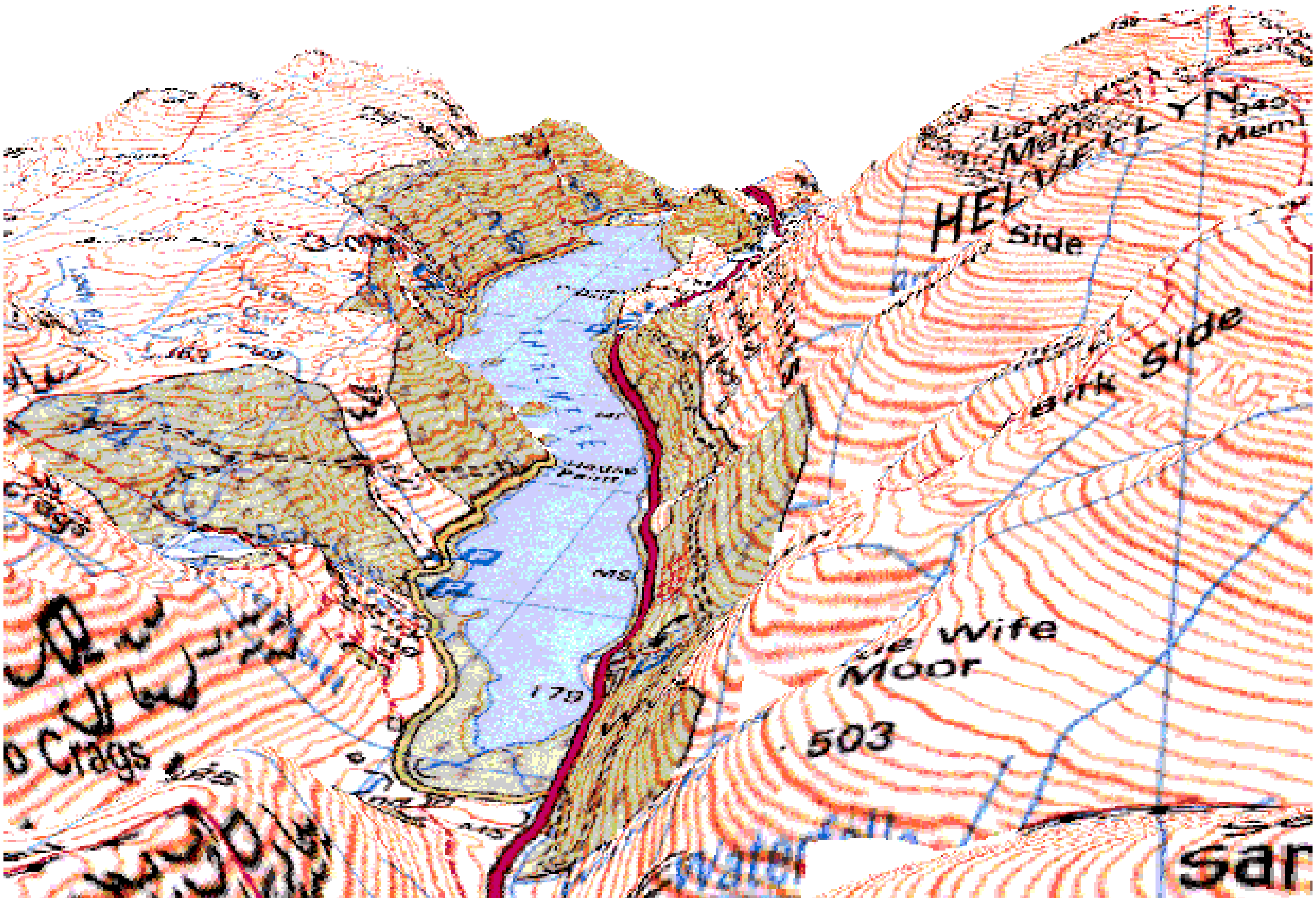
The Problem - Local vs. Global Minima

- All of the previous methods are *Hill-Climbing* strategies. Once you're on the top of the nearest hill, you can't get any higher.

- Q: How do you find the highest point?

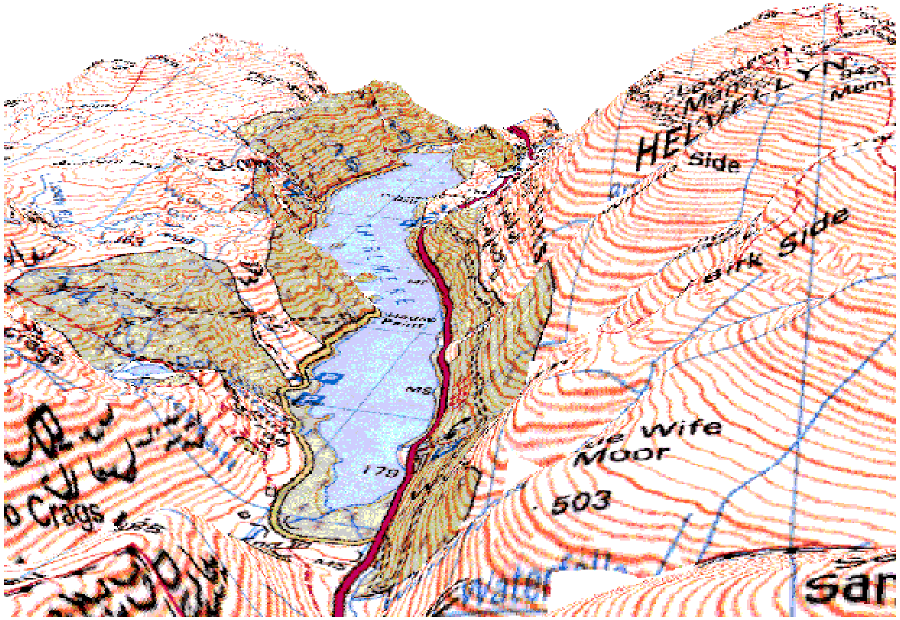


Back to The Map Analogy



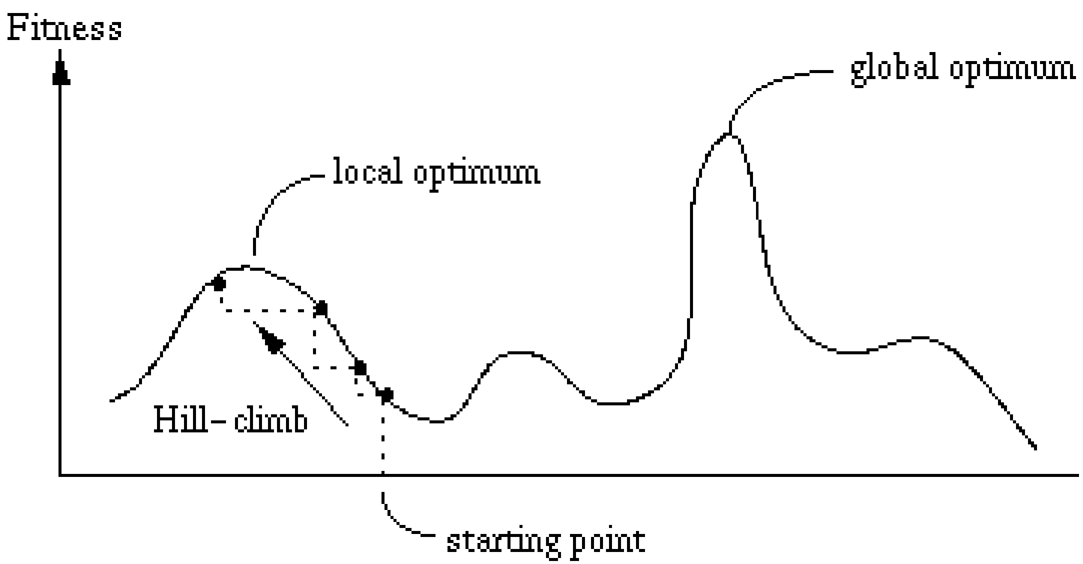
Finding Global Minima - Random Search

- Choose points randomly in the configuration space. Unintelligent, and rarely used by itself.
- However, it is useful for comparing with other methods to see if they're working.
- Of course, over a long enough time the random search is guaranteed to find the optimum solution!



Finding Global Minima - Stochastic Hill-Climbing

- Instead of just climbing up the nearest hill and you can also make random steps, retaining the move if the fitness is improved.
- Easy to implement and fast, but is ‘noisy’ if there are many small peaks.



Simulated Annealing (Metropolis, 1953)

- Analogy with thermodynamics - a liquid cooled slowly forms a large crystal where the atoms are nearly at their minimum (optimum) energy state.
- Key to optimisation process is slow cooling, where there is time for movement to the lowest energy state - this is annealing.
- The previous methods correspond to quenching.

Simulated Annealing – Principles (Metropolis, 1953)

- Boltzmann distribution gives probability of system being in a state of energy E,

$$P(E) \sim \exp\left(\frac{-E}{kT}\right)$$

- Simulated annealing gives probability of transition from energy E1 to E2 with probability

$$p = \exp\left[\frac{-(E_2 - E_1)}{kT}\right]$$

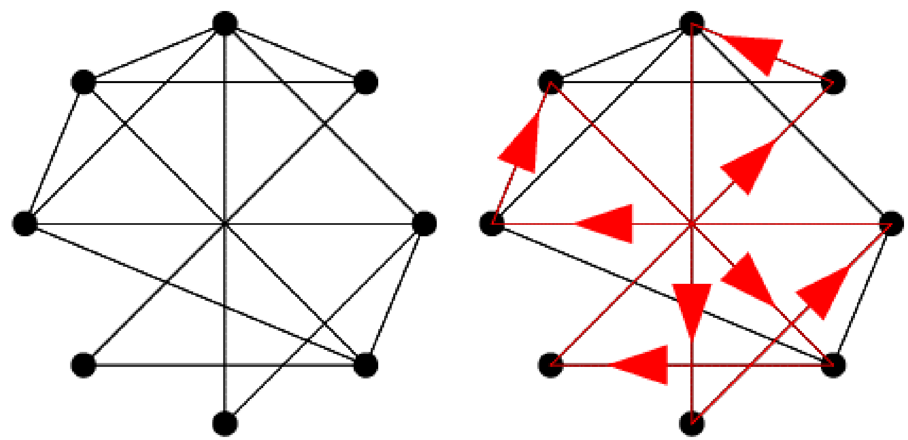
Simulated Annealing – Implementation (Metropolis, 1953)

- The algorithm uses the following elements:
  - 1. A definition of the configuration space.
  - 2. A generator of random changes in the configuration. These are the energy ‘options’ presented to the system.
  - 3. An objective function E (analog of energy) to minimise.
  - 4. A control parameter T (analog of temperature) and an annealing schedule - how large and how often the downward steps in T are.
- High T gives high P of moving to a worse state - explores configuration space.
- Low T gives settling to final optimum.

Simulated Annealing: The Travelling Salesman Problem

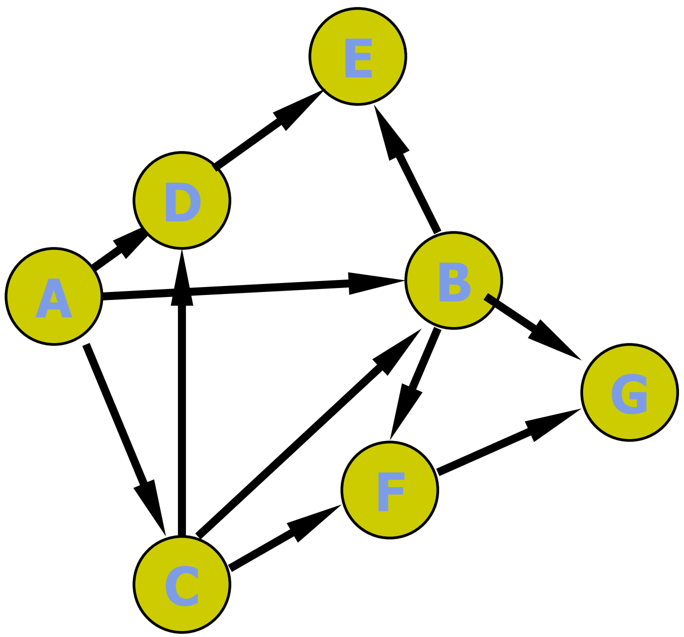
- A classic problem in optimisation - how does the salesman travel the least distance while only visiting each city only once?
  - Shortest ***Hamiltonian Cycle***
- Start with an initial path and perform changes to reduce objective function.
- With infinitely slow cooling the shortest path is definitely found.
- This class of problem is ***NP-Complete***
  - NP: Polynomial Time
  - The only sure solution is exhaustive search.

$$E = L \equiv \sum_{i=1}^{N} \sqrt{\left(x_i - x_{i+1}\right)^2 + \left(y_i - y_{i+1}\right)^2}$$

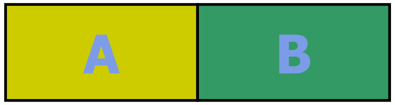


An Aside - DNA Computers

- Adleman (1994) showed that DNA computers can solve complex problems requiring extremely parallel processing.
- The Travelling Salesman problem is one of these so-called NP-complete problems.
- DNA pieces representing possible steps are allowed to combine in random sequences.

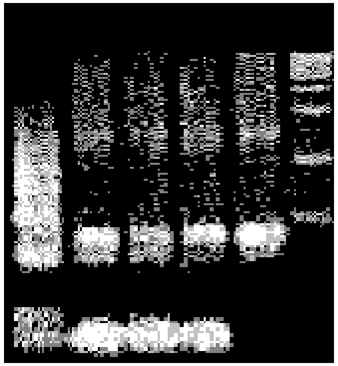


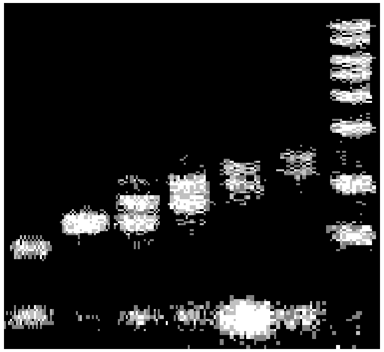
Hamiltonian Cycle Problem - going through every vertex with the shortest path



An Aside - DNA Computers

- Every possible sequence is tried.
- Sequences of short length (low M) and with the right start and end points are selected for amplification (PCR). The final product shows the shortest path.
- This method scales to large numbers of steps which cannot be solved on ordinary computers.





All possible paths **Shortest Path**

Genetic Algorithms (Holland, 1975)

- The concept is a Population of points in configuration space.
- Each point $\underline{P}$ is represented by a Gene
  - a binary representation which can be decoded to give the Phenotype – i.e. the Point $\underline{P}$ (the ‘design’)

- The Population is allowed to Evolve with interaction between the individuals.
- Eventually the population will Converge to a fitter region of the configuration space.

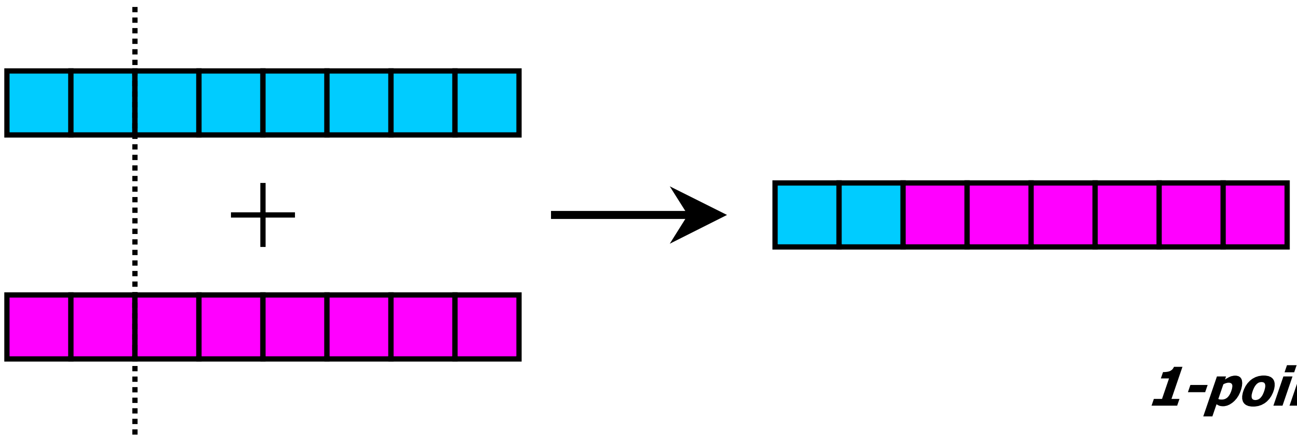
- This is how Nature does it!

Basic Genetic Algorithms

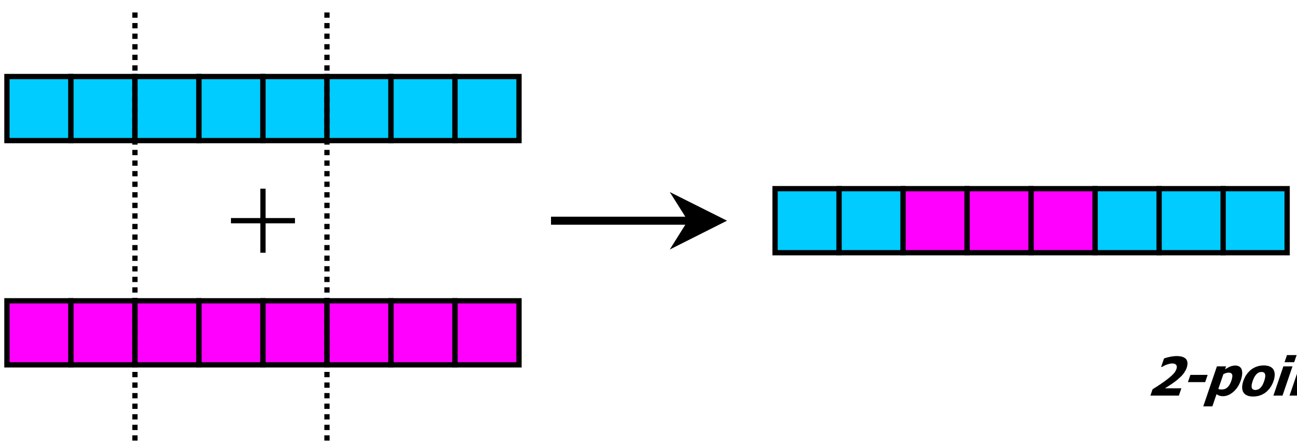
- Evolution of the population proceeds through the following steps:

Basic Genetic Algorithms - Reproduction

- Reproduction proceeds through crossover:



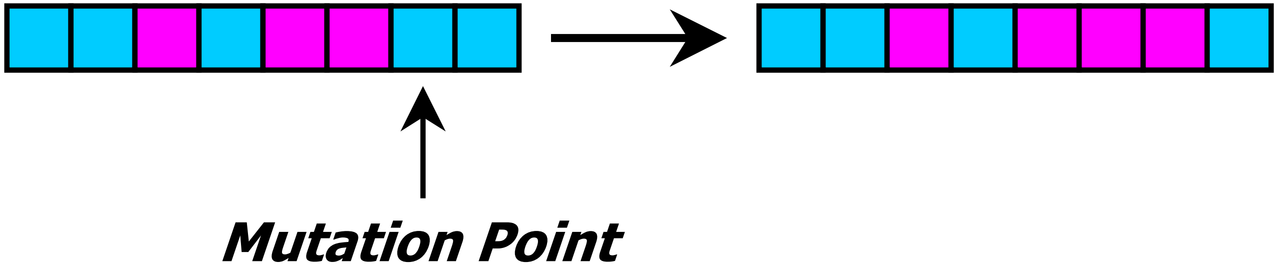
1-point



2-point

Basic Genetic Algorithms - Mutation

- Mutations are characterised by a Mutation Rate.



Basic Genetic Algorithms - Selection

- Selection can proceed in various ways:
  - 1. Only the best children are kept (no parents kept).
  - 2. Parents and children are ranked together, and only the best are kept.
  - 3. Each child is compared to the parent most like it (using the Hamming Distance), which it replaces if it is better - This method is called Niching.
    - Hamming distance is number of different bits – a distance measure

- The method of selection is important as it is obviously non-stochastic. Selection gives pressure toward fitter regions of configuration space.

Basic Genetic Algorithms - Convergence

- The selection procedure and the mutation rate are important for determining how fast the population converges to a particular region of configuration space.
- The convergence rate determines how much ‘variety’ is tried.
- Strong analogy with Simulated Annealing technique, and with damping and excitation in phase space.
- Selection is analogous to damping, mutation is analogous to noisy excitation.

Basic Genetic Algorithms - Why they Work

- Theoretical foundations well-established. Based on the idea of a Schema - a section of the gene that codes for a particular aspect of the phenotype.

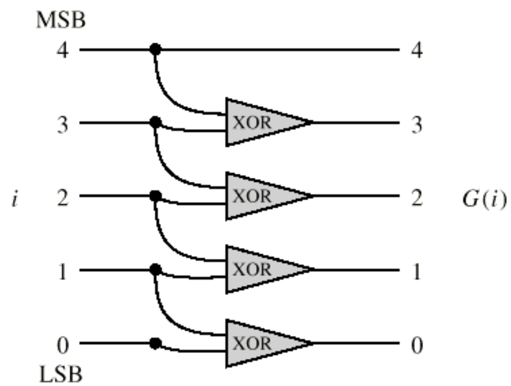
- Schema Theorem:
  - ‘Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm’

- To get the best optimisation, the order of bits in the gene representation should correspond to efficient schemas.
  - i.e. Need to code in a good way.

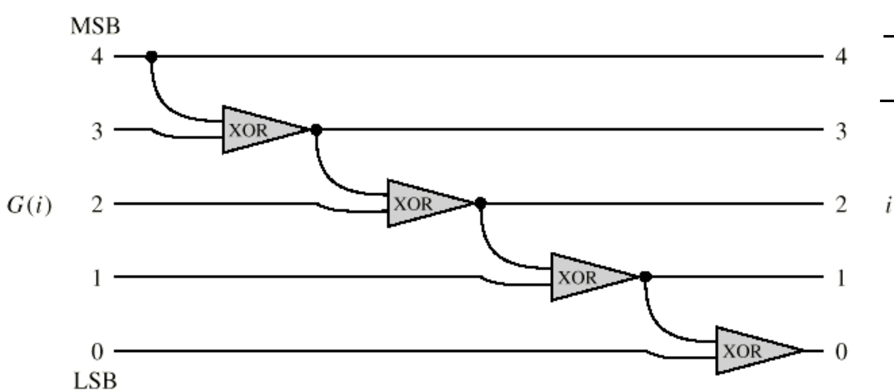
Genetic Algorithms - Dealing with Numbers

- Numeric parameters can be represented in binary form and encoded into the gene.
- However, ordinary binary representation means that changing one bit has a larger effect than changing another bit.
- This is overcome using the Gray Code form of the binary number.
  - A change in any single bit is equivalent to a change in any other - consecutive numbers have a Hamming Distance of one.

Gray Coding



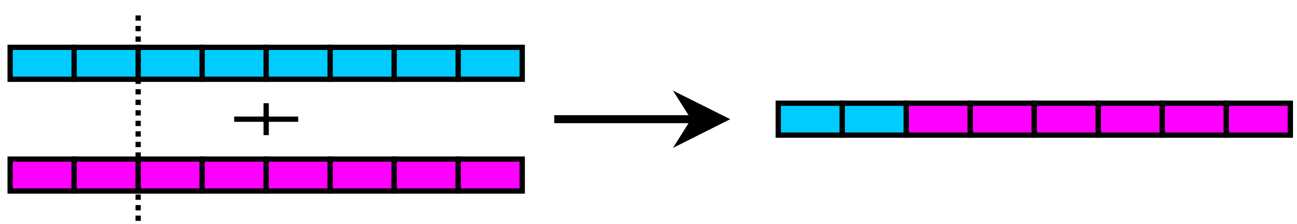
(a)



(b)

Integer	*Binary Code*	*Gray Code*
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

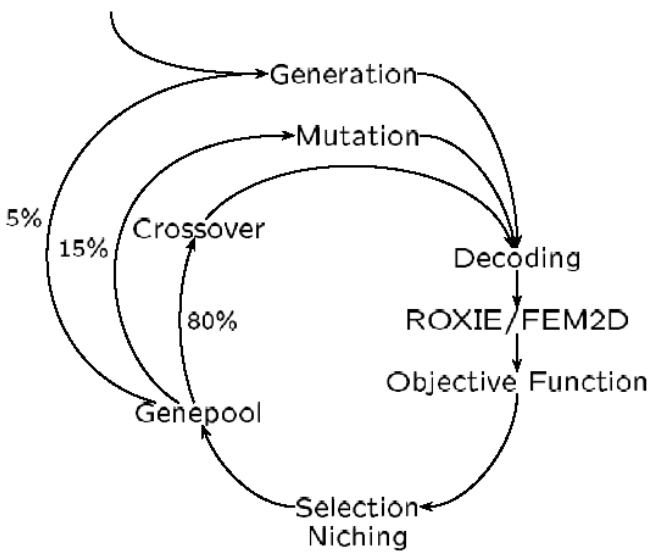
Neighbouring Gray codes have a Hamming distance of 1.



Example: LHC Dipole Yoke Design

- Russenschuck (1998) used genetic algorithms to optimise LHC dipole field quality by changing coil and yoke distributions.

$$
\begin{array}{cccc}
\varphi & \alpha & N_c & I \\
\downarrow & \downarrow & \downarrow & \downarrow \\
10011 & 01001 & 101 & 11011 \\
\downarrow & \multicolumn{2}{c}{\text{encoding}} & \downarrow \\
11010 & 01101 & 111 & 10110 \\
 & \multicolumn{2}{c}{\downarrow} & \\
 & \multicolumn{2}{c}{\text{Genetic Algorithm}} & \\
 & \multicolumn{2}{c}{\downarrow} & \\
01011 & 11101 & 011 & 01011 \\
\downarrow & \multicolumn{2}{c}{\text{decoding}} & \downarrow \\
01101 & 10110 & 010 & 01101 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\varphi & \alpha & N_c & I \\
 & \downarrow & \downarrow & \\
 & \multicolumn{2}{c}{\text{new designs}} &
\end{array}
$$



Example: LHC Dipole Yoke Design

- Surprising results obtained. Alternatives found to previous 5 block designs with improved field quality.

- Population size of 60 used. Gene length of between 50 and 60 bits.

- Similar design process performed for LHC quadrupoles.

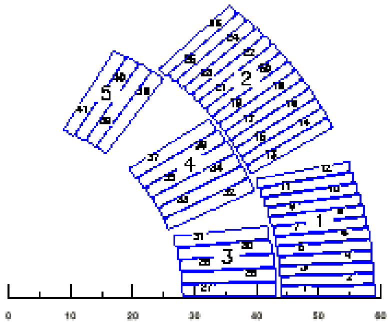


Fig. 4. Coil cross-section for the 5 block (41 turns) design (VY)

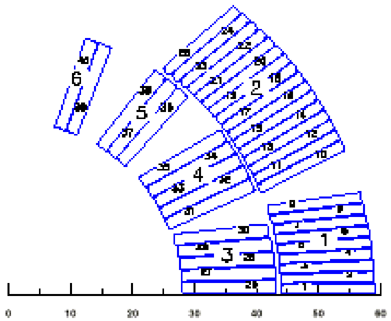


Fig. 5. Coil cross-section for the 6 block (40 turns) design (V6-1)

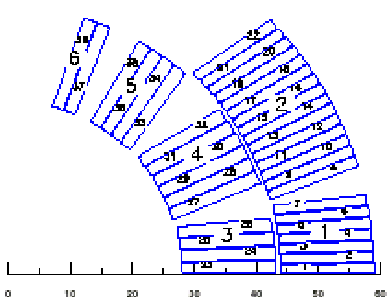
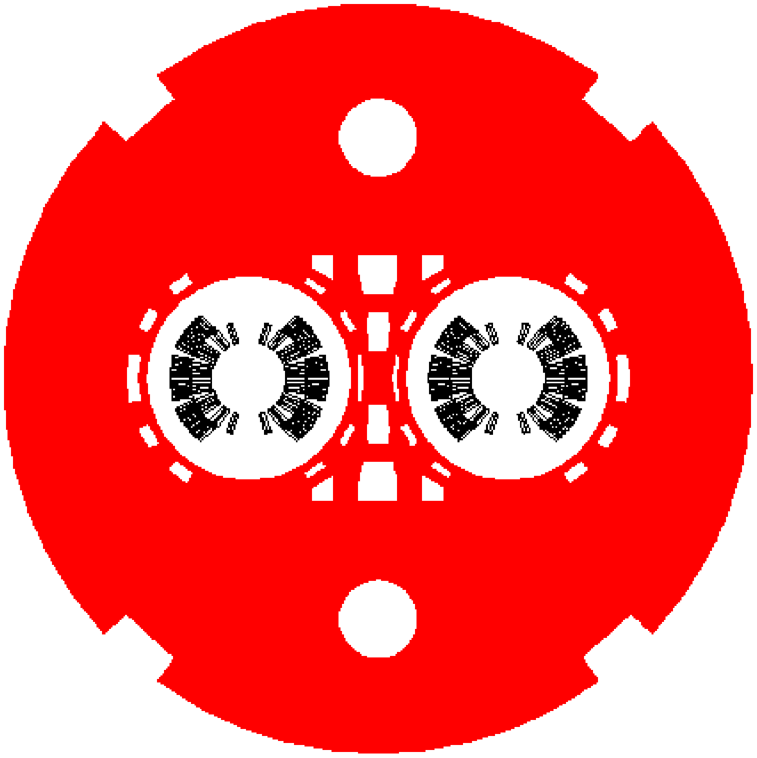


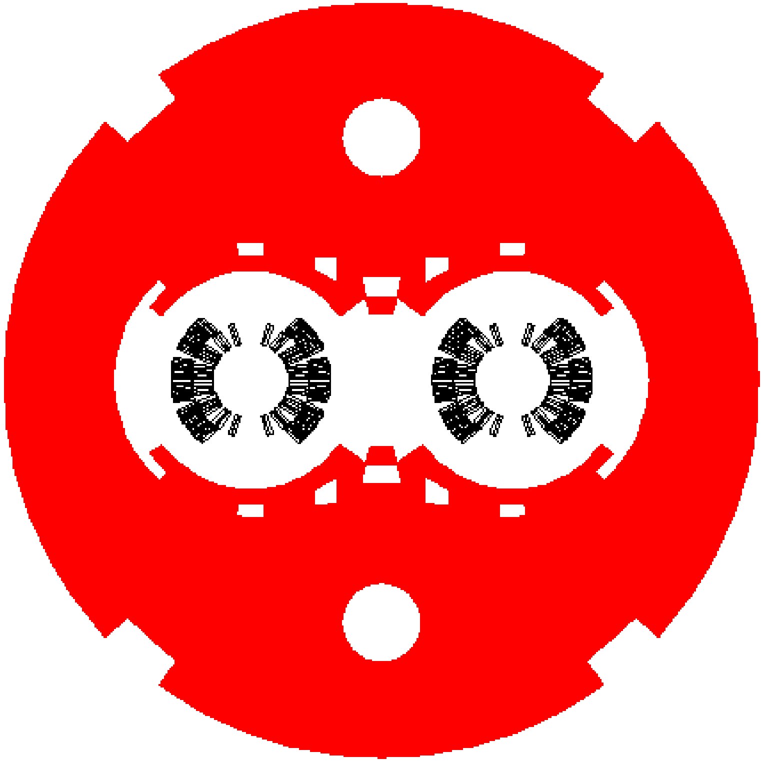
Fig. 6. Coil cross-section for the 6 block (38 turns) design (V6-3)

Example: LHC Dipole Yoke Design

- Optimising the distribution of the yoke material



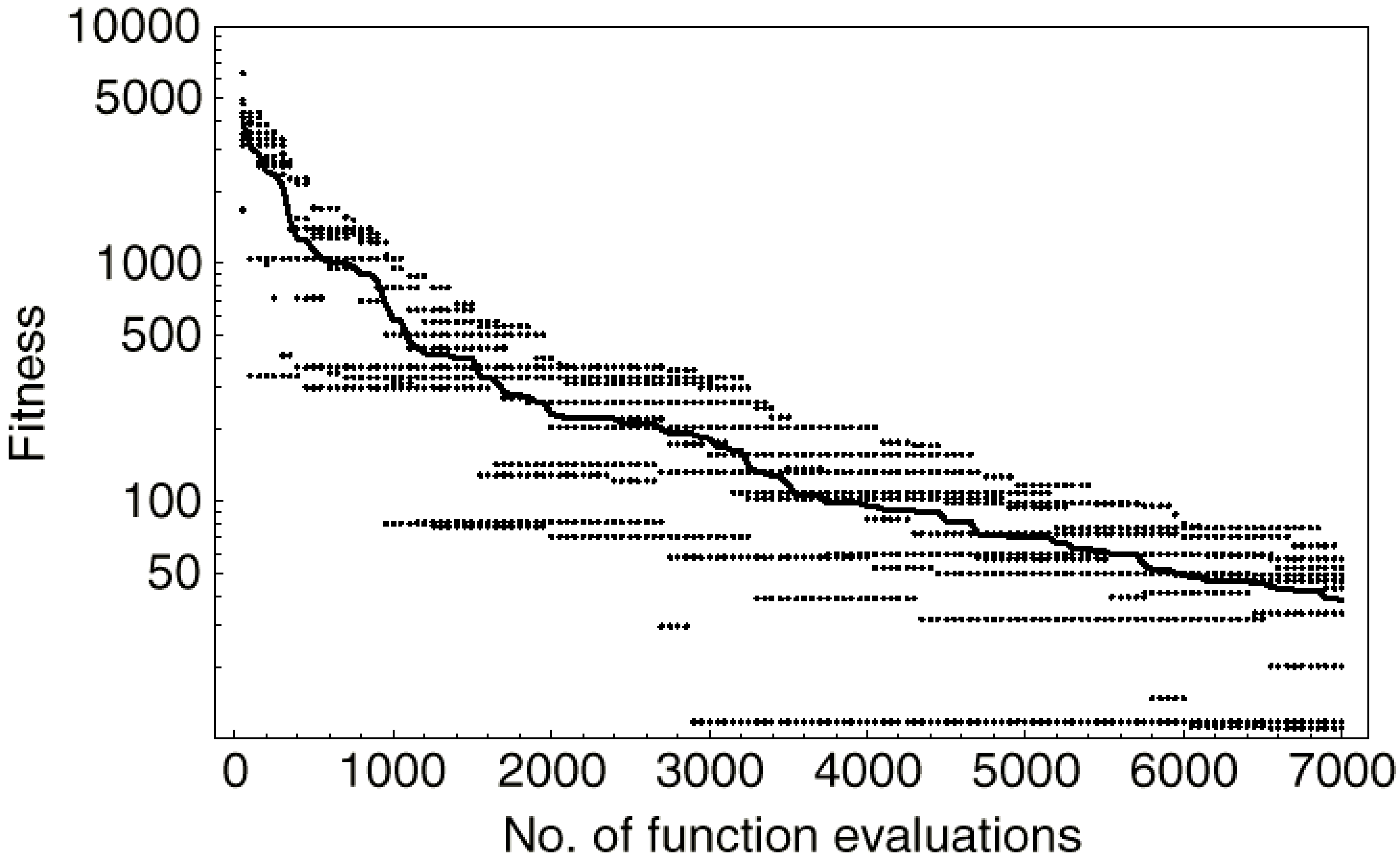
Best at one energy (injection)



Best over energy range

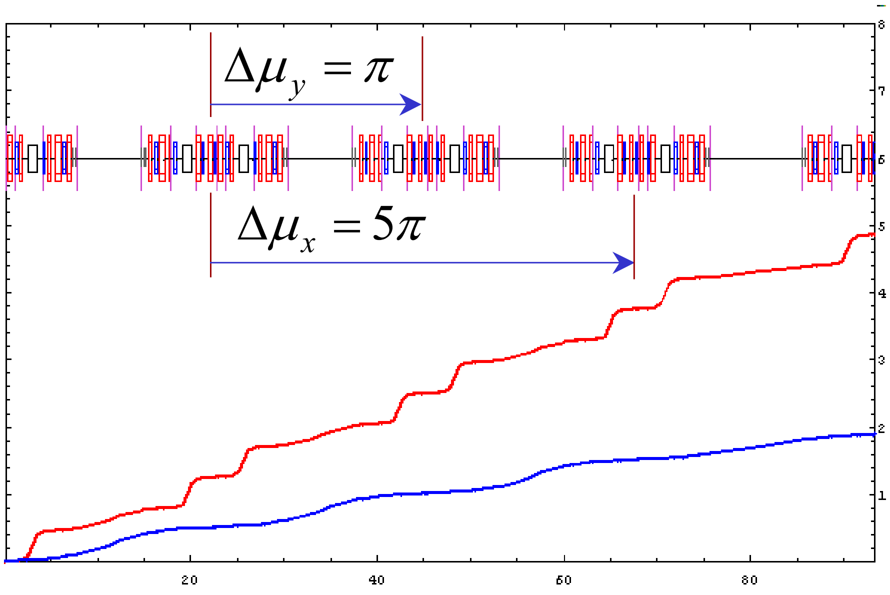
Example: LHC Dipole Yoke Design

- Convergence to a fit population:

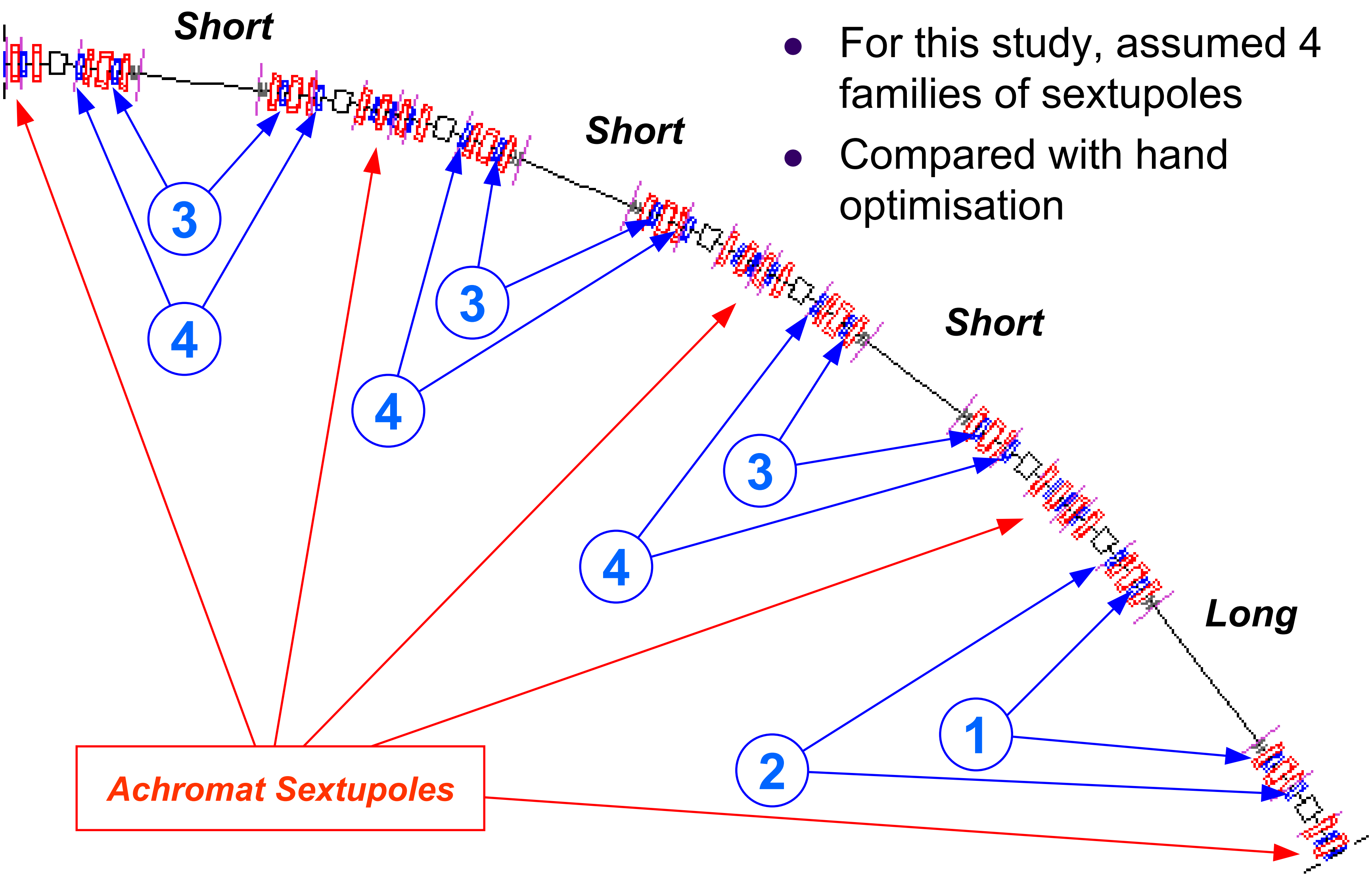


Example: DIAMOND Storage Ring Design

- Partial cancellation of chromatic sextupoles over 4 cells using appropriate phase advances
- 4 families of harmonic sextupoles to achieve full cancellation
- AP-SR-rpt-062 and -063 describe in detail



Sextupole Families



- For this study, assumed 4 families of sextupoles
- Compared with hand optimisation

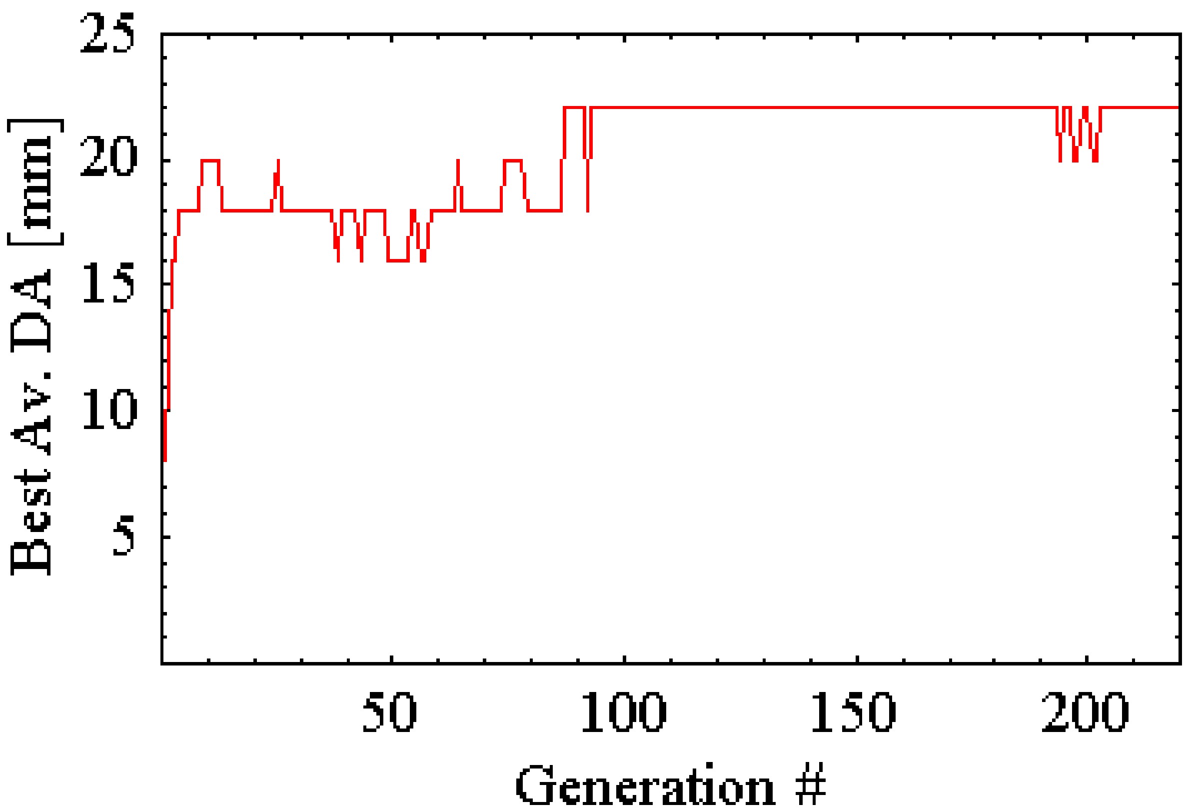
Defining the Objective Function

- Calculating the full dynamic aperture too costly in time
- Therefore define Objective Function *F* by:
  - Tracking 32 turns, on-momentum, in 1-D x and y
  - Find amplitude in x and y where stability <32 turns
  - Mean value is *F*.

- Need to trust the objective function is related to the real dynamic aperture (on- and off-momentum) that we are interested in
  - i.e. Trust it is a real ‘Quality Factor’

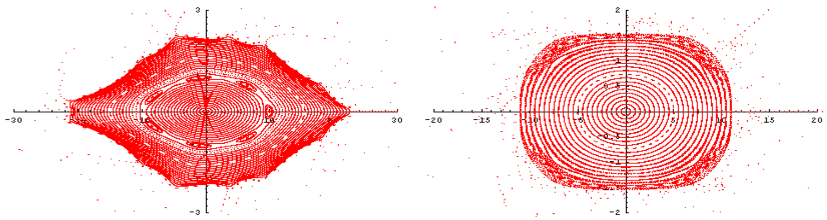
Implementation of Genetic Algorithm

- Use classical GA
- 4 sextupole strengths 36 bit string representation
- Population = 100
- No. of generations > 200

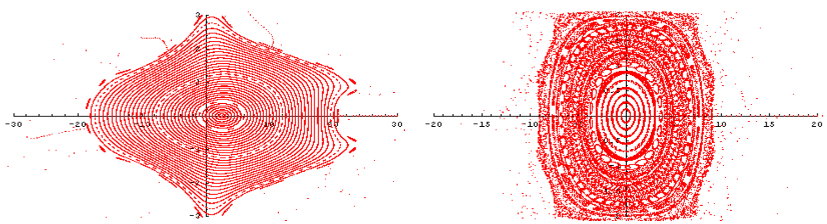


Comparison with Hand Optimisation

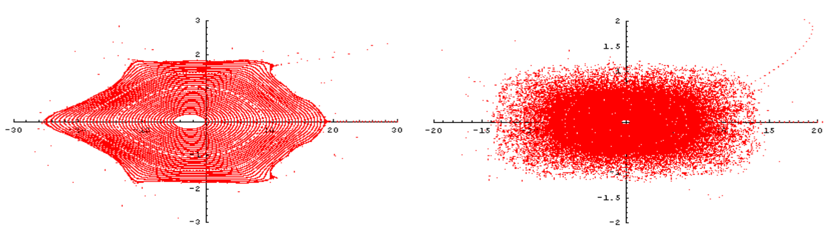
By Hand



On-momentum

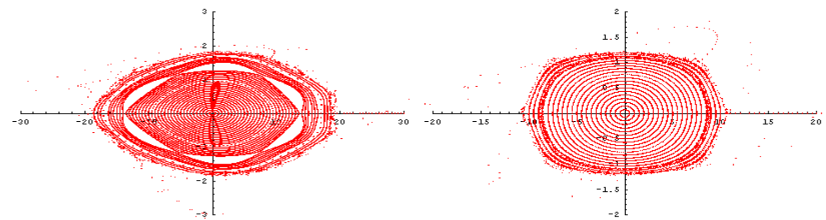


+4%

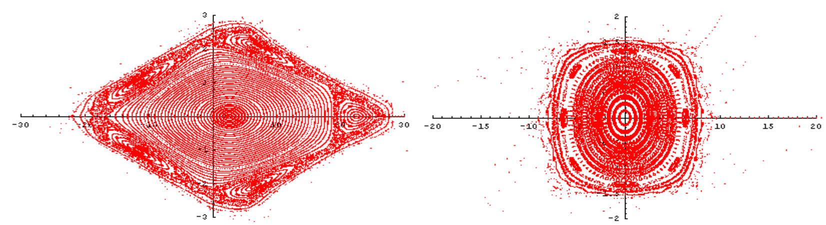


-4%

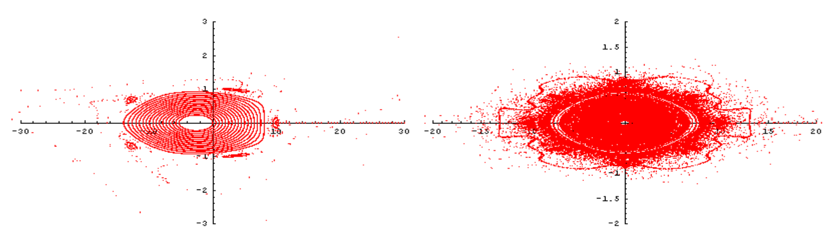
Genetic



On-momentum



+4%



-4%

Quality Factors and All That

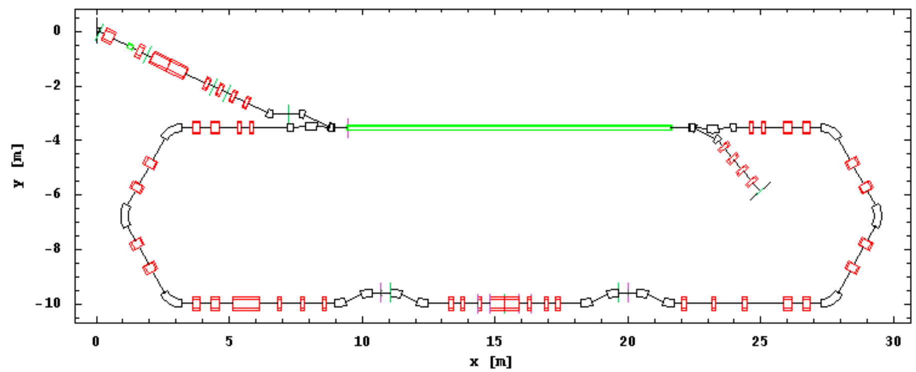
- We can see that on-momentum, the performance of the genetic optimisation is broadly similar.
  - Remember, there was no ‘intelligence’ behind the GA – it started from completely random choices of sextupole settings
- Off-momentum the performance is similar but maybe not so good.
  - *We* can see differences, the GA can’t
- **Moral** - make sure the Objective Function has the following properties:
  - Uses a true ‘Quality Factor’
  - Contains all the factors we care about

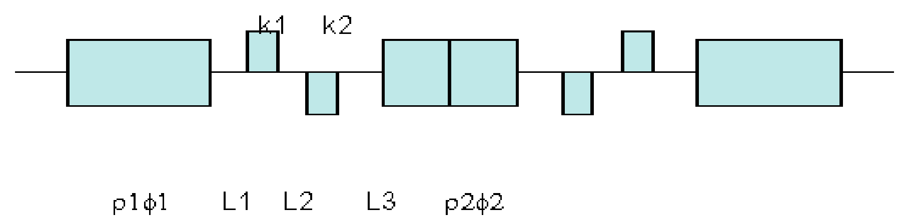
Evolutionary Algorithms/Programs and Hybrid Systems

- Often it is inconvenient to binary code numeric parameters. Instead perform ‘genetic-like’ operations on parameter sets
- Many different ways of doing this - called Evolutionary Algorithms and other names.
- Theoretical basis poor - but they work.
- Hybrid Systems utilise GA or EA methods, and use traditional optimisation (e.g. Simplex) to fine-tune the solution.

Example – Finding an Isochronous Arc

- 4GLS ERLP requires an isochronous arc
- We can define the arc by a set of assumed parameters:
  - Energy
  - Total Deflection Angle
  - Dipole and Quadrupole Lengths
  - Outer Quadrupole Spacings
- Then have free parameters:
  - L_3
  - B_1
- B_2, k_1, k_2, L_1, L_2, depend on these for isochronous solution





First-Order Isochronous Equations

$$R_{56} = \int_{s_1}^{s_2} \frac{D(s)}{\rho(s)}\,\mathrm{d}s \qquad \begin{aligned} R_{56}|_{\mathrm{outer}} &= \rho_1(\phi_1 - \sin\phi_1) \\ R_{56}|_{\mathrm{central}} &= D_j \sin(\phi_2/2) - \rho_2 D'_j[\cos(\phi_2/2) - 1] + \rho_2[\phi_2 - \sin(\phi_2/2)] \end{aligned}$$

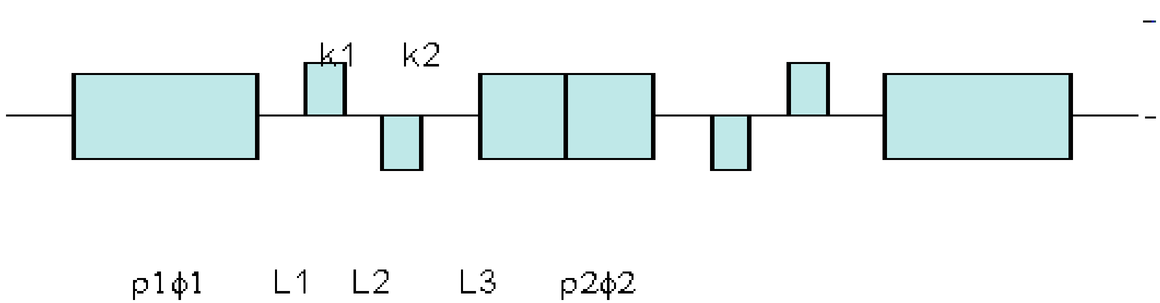
$$\begin{aligned} L_1 &= a\frac{C_2 q_1}{C_1 q_2}(L_3 - \frac{D_j}{D'_j} + q_2) - l + q_1 \\ L_2 &= q_1 - q_2 + \frac{b}{L_3 - D_j/D'_j + q_2}, \end{aligned}$$

$$\begin{aligned} l = \rho_1 \tan(\phi_1/2), \qquad & a = -D'_j/\sin(\phi_1), \\ b = \frac{q_2}{C_2}(\frac{q_2}{C_2} + \frac{q_1}{aC_1}), \quad & q_i = \frac{C_i}{S_i\sqrt{k_i}}\ (i = 1 \text{ or } 2), \\ C_1 = \cos(l_q\sqrt{k_1}), \qquad & S_1 = \sin(l_q\sqrt{k_1}), \\ C_2 = \cosh(l_q/\sqrt{k_2}), \qquad & S_2 = \sinh(l_q\sqrt{k_2},) \end{aligned}$$

$$\rho_1 = \frac{E}{B_1 ce}, \quad \rho_2 = -\frac{NEl_m}{2ceNl_mB_1 + E\theta_t},$$

$$\phi_1 = \frac{cel_mB_1}{E}, \quad \phi 2 = \frac{\theta_t}{N} - \frac{2cel_mB_1}{E},$$

$$B_2 = \frac{E\theta_t}{ceNl_m} - 2B_1.$$



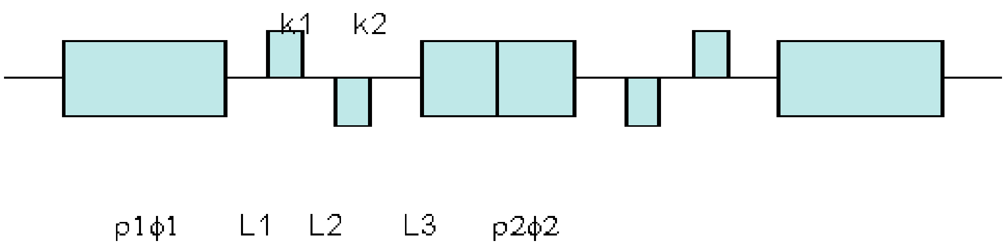
Optimisation by Direct Search

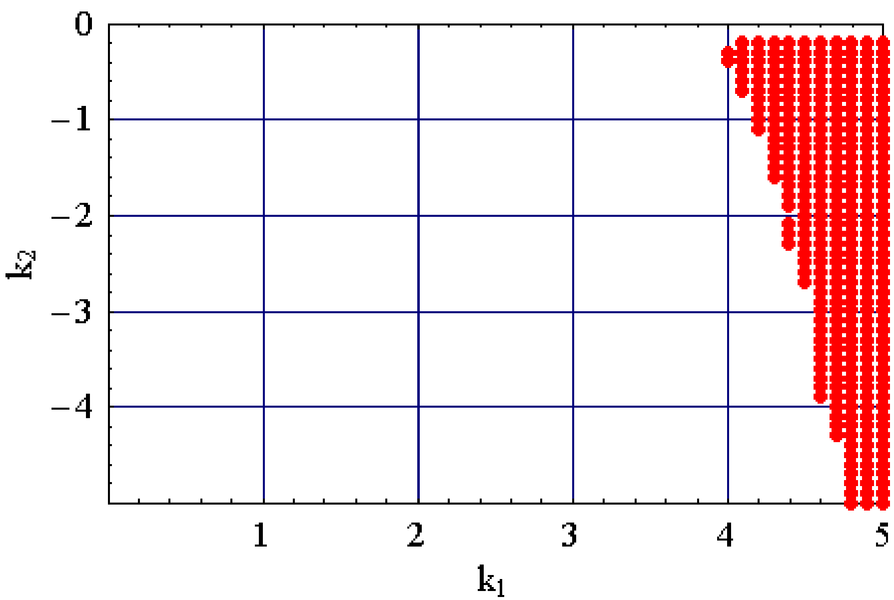
- Some values of the free parameters have no solution
- We also want a global view of the dependencies
- Objective function is fast to calculate
  - e.g. Minimum of β_x, β_y

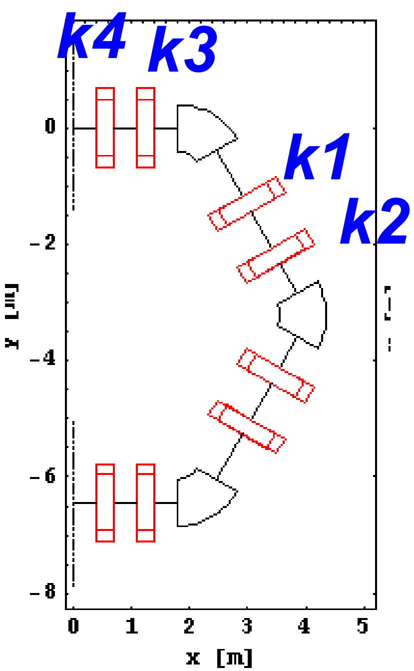
- Direct search is therefore a good method
  - Explicitly calculate ALL solutions (over some grid of a particular step size)
  - We then know we have the best solution

Isochronous TBA Results

- A complete scan of all possible solutions has been carried out:
  - $0 \text{ m} < L < 1 \text{ m}$
  - k_1, k_2 up to 5 m^{-1}
  - Equal B_1, B_2
  - Assumed some lengths

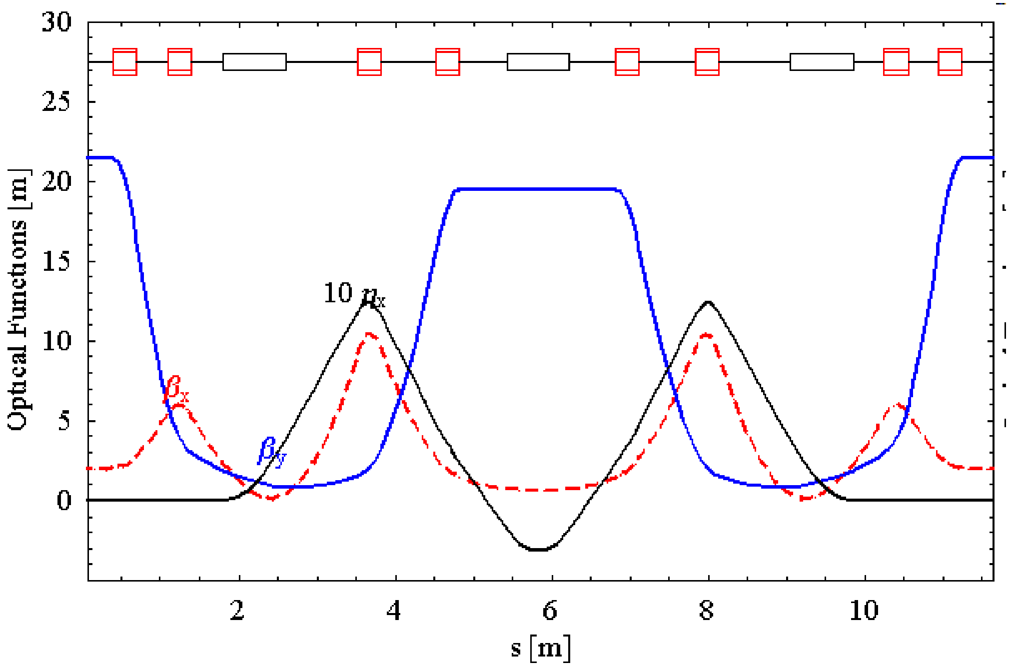


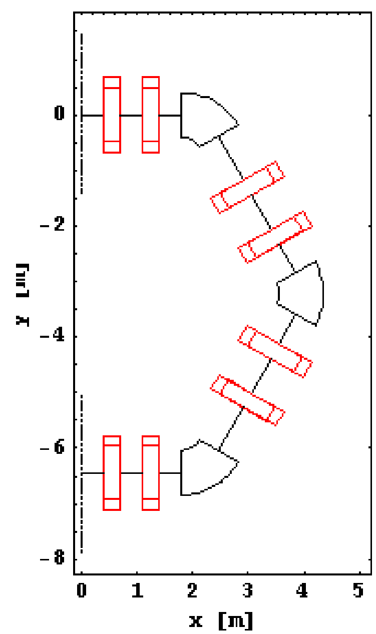




Finding the Optics Solution

- For each Iso solution, scan k_3/k_4
- Choose overall smallest betatron sum
- Starting solution for some field
- Also done by scanning





Which Optimisation Method Is Best?

- Genetic algorithms have distinct advantages over classical single-point optimisation techniques for particular classes of problems:
  - 1. Best area of configuration space is not known
  - 2. Many peaks/discontinuous Objective Function
  - 3. Best solution not required - ‘good enough’ needed
- Hybrid solutions are popular, combining several methods.
- No particular algorithm is best in the general case.

No Free Lunch Theorem (Wolpert and Macready, 1995)

- Important general theorem of search algorithms:
  - ‘All algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions.’
- In other words, if algorithm A outperforms algorithm B for some cost functions, then there must exist as many functions where B outperforms A.
- Corollary:
  - The Algorithm must be matched to the Problem.